The hitchhiker's guide to decidability and complexity of equivalence properties in security protocols*

Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina

Inria Nancy Grand-Est & LORIA

Abstract. Privacy-preserving security properties in cryptographic protocols are typically modelled by observational equivalences in process calculi such as the applied pi-calulus. We survey decidability and complexity results for the automated verification of such equivalences, casting existing results in a common framework which allows for a precise comparison. This unified view, beyond providing a clearer insight on the current state of the art, allowed us to identify some variations in the statements of the decision problems—sometimes resulting in different complexity results. Additionally, we prove a couple of novel or strengthened results.

Keywords: Formal verification · Cryptographic protocols · Complexity.

1 Introduction

Symbolic verification techniques for security protocols can be traced back to the seminal work of Dolev and Yao [38]. Today, after more than 30 years of active research in this field, efficient and mature tools exist, e.g. PROVERIF [15] and TAMARIN [50] to only name the most prominent ones. These tools are able to automatically verify full fledged models of widely deployed protocols and standards, such as TLS [14,34], Signal [47,29], the upcoming 5G standard [11], or deployed multi-factor authentication protocols [45]. We argue that the development of such efficient tools has been possible due to a large amount of more theoretical work that focuses on understanding the precise limits of decidability and the computational complexity of particular protocol classes [39,41,49,40,30,46].

The abovementioned results extensively cover verification for the class of *reachability* properties. Such properties are indeed sufficient to verify authentication properties and various flavors of confidentiality, even in complex scenarios with different kinds of compromise [10]. Another class of properties are *indistinguishability* properties. These properties express that an adversary cannot distinguish two situations and are conveniently modelled as *observational equivalences* in a cryptographic process calculus, such as the applied pi calculus. Such equivalences can indeed be used to model strong flavors of secrecy, in terms of non-interference or as a "real-or-random" experiment.

* The research leading to these result has received funding from the ERC under the EU's H2020 research and innovation program (grant agreements No 645865-SPOOC), as well as from the French ANR project TECAP (ANR-17-CE39-0004-01). Itsaka Rakotonirina benefits from a Google PhD Fellowship.

Equivalences are also the tool of choice to model many other privacy-preserving properties. Such properties include anonymity [3], unlinkability properties [6,42], as well as vote privacy [37] to give a few examples. Equivalence properties are inherently more complex than reachability properties, and both the theoretical understanding and tool support are more recent and more brittle. This state of affairs triggered a large amount of recent works to increase our theoretical understanding and improve tool support.

In this paper we give an extensive overview of decidability and complexity results for several process equivalences. In particular, in this survey we give a unified view, allowing us to highlight subtle differences in the definitions of the decision problems across the literature (such as whether the term theory is part of the input or not) as well as the protocol models. Typically, models may vary in whether they allow for a bounded or unbounded number of sessions, the support of cryptographic primitives, whether they support else branches (i.e. disequality tests, rather than only equality tests), and various restrictions on non-determinism. Additionally, our technical report [25] contains full proofs of all results that are novel or that required additional arguments to make up for the differences in stating the problem compared to the original work. All the results are summarised in Table 1, and we identify several open questions. Delaune and Hirschi [36] also survey symbolic methods for verifying equivalence properties. However, they mainly discuss tool support whereas we focus on computational complexity.

2 Model

We will model protocols as processes in the applied pi calculus, and cryptographic primitives are modelled using terms equipped with rewrite rules. We assume the reader is familiar with these notions and only recall them briefly and informally. A fully-detailed model can be found in the technical report [25].

Cryptographic primitives. As usual in symbolic protocol analysis we take an abstract view of cryptography and model the messages exchanged during the protocol as *terms* built over a set of function symbols each with a given arity. Terms are then either atomic values or function symbols applied to other terms, respecting the function's arity. Atomic values are either *constants*, i.e., function symbols of arity 0 or *names*. Constants, sometimes referred to as public names, model public values, such as agent identities or protocol tags. Names model secret values, such as keys or nonces, and are a priori unknown to the adversary. We assume an infinite set of constants and names.

Example 1. For example the encryption of a plaintext m with a key k using a symmetric encryption scheme senc is modelled by the term $\mathsf{senc}(m, k)$. △

The functional properties of the symbols are modelled by an *equational theory*. In this work we restrict ourselves to equational theories that can be oriented into a *convergent rewriting system*. This also implies that any term t has a unique normal from $t{\downarrow}$.

Example 2. The rewrite rule $\mathsf{sdec}(\mathsf{senc}(x, y), y) \to x$ defines the behaviour of the encryption scheme: one can decrypt (apply sdec) a ciphertext $\mathsf{senc}(x, y)$ with the corresponding key y to recover the plaintext x. This behaviour is idealised by the absence of other rules for senc and sdec, modelling an assumption that no information can be

extracted from a ciphertext except by possessing the decryption key. Similarly, asymmetric encryption can be modelled by the rewrite rule $\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) \to x$ where pk is a symbol of arity 1 modelling public keys. Such rewrite rules can express a broad range of other primitives like pairs ($\mathsf{fst}(\langle x, y\rangle) \to x$ and $\mathsf{snd}(\langle x, y\rangle) \to y$), hash functions (no rewrite rule) or randomised encryption (adding an additional argument to senc to explicitly represent the randomness). △

In this survey we call a *theory* the set of non-constant function symbols together with a rewriting system. Two classes of theories are particularly important for our results. The first is the class of *subterm convergent* theories [2,12,16,22,28,31], defined by a syntactic criterion on rewriting rules $\ell \to r$ requiring that r is either a strict subterm of ℓ or a ground term in normal form. The second is the class of *constructor-destructor* theories [16,20,22], partitioning function symbols into constructor (used to build terms) and destructors (only used in rewrite rules). In constructor-destructor theories any rewrite rule $\ell \to r$ is such that $\ell = d(t_1, \ldots, t_n)$ where d is a destructor and $t_1, \ldots, t_n, r$ do not contain any destructor. Moreover, we assume a *message* predicate $\mathsf{msg}(t)$ which holds if $u{\downarrow}$ does not contain any destructor symbol for all subterms u of t, i.e., all destructor applications in t succeeded yielding a valid message. This predicate is used to restrict to protocols that only send and accept such well-formed messages.

Protocols. Protocols are defined using processes in the applied pi calculus. The syntax of protocols is defined by the grammar of *processes*:

$$P, Q ::= \qquad 0 \qquad \mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \qquad c(x).P \qquad \overline{c}\langle u\rangle.P \qquad P \mid Q$$

Intuitively the 0 models a terminated process, a conditional $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$ executes either P or Q depending on whether the terms $u{\downarrow}$ and $v{\downarrow}$ are equal, and $P \mid Q$ models two processes executed concurrently. The constructs $c(x).P$ and $\overline{c}\langle u\rangle.P$ model, respectively, inputs and outputs on a communication channel c. When the channel c is known to the attacker, e.g. when it is a constant, executing an output on c adds it to the adversary's knowledge and inputs on c are fetched from the adversary possibly forwarding a previously stored message, or computing a new message from previous outputs. Otherwise the communication is performed silently without adversarial interferences. To model an unbounded number of protocol sessions we also add the two constructs

$$P, Q ::= \qquad \mathsf{new}\ k.P \qquad !P$$

The replication $!P$ models an unbounded number of parallel copies of P, and $\mathsf{new}\ k.P$ creates a fresh name k unknown to the attacker; in particular $!\mathsf{new}\ k.P$ models an unbounded number of sessions, each with a different fresh key. The fragment of the calculus without replication is referred to as *finite* or *bounded*. Another notable subclass is the original pi-calculus [48], referred to as the *pure* fragment, that can be retrieved with the empty theory (only names, constants and an empty rewrite system).

Semantics in an adversarial environment. The behaviour of processes is formalised by an operational semantics. The detailed presentation differs from one work to another [1,21,22] and we only give a high-level overview here. It takes the form of a transition relation $(P, \Phi) \xrightarrow{\alpha} (P', \Phi')$ on *configurations* (P, Φ) where P is the process to be

executed, and Φ is called a *frame* and records the attacker knowledge. A frame is a substitution of the form $\{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\}$ where t_i are previous outputs and ax_i are special variables called *axioms* that serve as handles to the adversary for building new terms. The label α of the transition step is called an *action* and is either

- an *unobservable action* τ which represents an internal action, such as the evaluation of a conditional or a communication on a private channel;
- an *input action* $\xi_c(\xi_t)$ where ξ_c (resp. ξ_t) represents the attacker's computation of the input's public channel (resp. of the term to be input), see *recipes* in the next section;
- an *output action* $\overline{\xi_c}\langle \mathsf{ax}_i \rangle$ where ξ_c is again the attacker's computation of the channel, and the underlying output term is added to the frame as axiom ax_i.

We refer to the technical report [25] for full details of the semantics but provide additional intuition through the following example. Suppose that an agent S wants to send a nonce N to a recipient R. Assuming S and R already share a secret k_s, S encrypts N and k_s with the public key of R, i.e. $\mathsf{pk}(k_R)$ where k_R is the corresponding private key. When receiving a message, R acknowledges the nonce only if the plaintext contains the shared secret. This is modelled by the following process:

$$\begin{array}{lll} P = S \mid R & \text{with} & S = \overline{c}\langle M \rangle \qquad \text{where } M = \mathsf{aenc}(\langle N, k_s \rangle, \mathsf{pk}(k_R)) \\ & \text{and} & R = c(x).\, \mathsf{if}\ \mathsf{snd}(\mathsf{adec}(x, k_R)) = k_s\ \mathsf{then}\ \overline{c}\langle \mathsf{ack} \rangle\ \mathsf{else}\ 0 \end{array}$$

with $k_s, k_R, N \in \mathcal{N}$ and $c \in \Sigma_0$. The 0 processes are omitted. The fact that the public key should be known to the attacker is modelled by a frame $\Phi_0 = \{\mathsf{ax}_0 \mapsto \mathsf{pk}(k_R)\}$. A "normal" execution of the protocol would be, with informal notations:

$$\begin{array}{rll} (S \mid R, \Phi_0) \xrightarrow{\overline{c}\langle \mathsf{ax}_1 \rangle} (0 \mid R, \Phi_1) & \text{with } \Phi_1 = \Phi_0 \cup \{\mathsf{ax}_1 \mapsto M\} & (1) \\ \xrightarrow{c(\mathsf{ax}_1)} (0 \mid \mathsf{if}\ \mathsf{snd}(\mathsf{adec}(M, k_R)) = k_s\ \mathsf{then}\ \overline{c}\langle \mathsf{ack} \rangle\ \mathsf{else}\ 0, \Phi_1) & & (2) \\ \xrightarrow{\tau} (0 \mid \overline{c}\langle \mathsf{ack} \rangle, \Phi_1) & & (3) \\ \xrightarrow{\overline{c}\langle \mathsf{ax}_2 \rangle} (0 \mid 0, \Phi_1 \cup \{\mathsf{ax}_2 \mapsto \mathsf{ack}\}) & & (4) \end{array}$$

Here the attacker is passive and only forwards messages. More precisely in transition (1), S sends M which is added to the frame as reference ax_1. This models that the attacker spies on the communication network and gets access to all messages sent on public channels like c. In transition (2) the attacker forwards M to R, i.e. inputs ax_1. Transition (3) is an internal test of R which leads to the final acknowledgement output (4). An active attacker would also have the capability of forging new messages and inserting them in the execution flow. For example transition (2) can be replaced by the input $\xrightarrow{c(\mathsf{aenc}(\langle a,b \rangle, \mathsf{ax}_0))}$ with $a, b \in \Sigma_0$: rather than forwarding M the attacker encrypts the pair of constants a, b with the public key of R (using reference ax_0) and sends it to R. In this modified execution the subsequent test would however fail. Finally let us mention that for constructor-destructor theories, all attacker-crafted terms must be valid messages, i.e. satisfy the predicate msg [22,27].

When defining security against an active attacker we quantify over all such transitions which means we consider all possible executions in an active adversarial environment. Thus even the bounded fragment yields an infinite transition system if the theory contains a non-constant function symbol (as this allows to build an unbounded number of messages).

3 Complexity of static equivalence (passive attacker)

3.1 Static equivalence

Attacker knowledge. As explained above, frames $\Phi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\}$ record the outputs t_i performed during the execution of a process. They therefore enable adversarial deductions as they aggregate: for example after observing a ciphertext and the decryption key, the attacker can also obtain the plaintext by decrypting. Formally we say that one can *deduce* all terms of the form $\xi\Phi\downarrow$ where ξ is called a *recipe* that is a term built from function symbols, axioms $\mathsf{ax}_i \in dom(\Phi)$, and constants. Recipes were mentioned in the previous section, in the operational semantics, as the way to specify attacker's computations. The fact that recipes cannot contain names models that names are assumed unknown to the adversary initially. For example in $\Phi = \{\mathsf{ax}_1 \mapsto \mathsf{senc}(t, k), \mathsf{ax}_2 \mapsto k\}$ the term t is deducible by the recipe $\mathsf{sdec}(\mathsf{ax}_1, \mathsf{ax}_2)$, regardless of k being a name (which is not allowed to occur directly in the recipe).

Indistinguishability. Some security properties against a passive attacker, i.e. a simple eavesdropper, can then be modelled as an observational equivalence of two frames: intuitively no equality test can be used to distinguish them. For example, in a protocol that outputs a sequence of messages $t_1, \ldots, t_n$, the "real-or-random" confidentiality of a key k can be modelled as the equivalence of

$$\Phi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n, \mathsf{ax} \mapsto k\} \quad \Psi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n, \mathsf{ax} \mapsto k'\}$$

where k' is a fresh key. More formally, two frames Φ, Ψ with same domain are *statically equivalent* when for all recipes ξ_1, ξ_2 we have that

$$\xi_1\Phi\downarrow = \xi_2\Phi\downarrow \iff \xi_1\Psi\downarrow = \xi_2\Psi\downarrow \ .$$

In constructor-destructor theories we also require that $\mathsf{msg}(\xi_1\Phi)$ *iff* $\mathsf{msg}(\xi_1\Psi)$, modelling an assumption that the adversary can observe destructor failures.

Example 3. If k, k' are names, $\Phi = \{\mathsf{ax} \mapsto k\}$ and $\Psi = \{\mathsf{ax} \mapsto k'\}$ are equivalent, capturing the intuition that random keys cannot be distinguished. However, for the constant 0, $\Phi = \{\mathsf{ax}_1 \mapsto \mathsf{senc}(0, k), \mathsf{ax}_2 \mapsto k\}$ and $\Psi = \{\mathsf{ax}_1 \mapsto \mathsf{senc}(0, k), \mathsf{ax}_2 \mapsto k'\}$ are not equivalent since $\xi_1 = \mathsf{sdec}(\mathsf{ax}_1, \mathsf{ax}_2)$ and $\xi_2 = 0$ are equal in Φ but not in Ψ. △

3.2 Complexity results

We study the following decision problem referred to as STATEQ:

> INPUT: A theory, two frames of same domain.
>
> QUESTION: Are the two frames statically equivalent for this theory?

General case. As rewriting is Turing-complete, unsurprisingly static equivalence is undecidable in general for convergent rewriting systems [2]. It is also proved in [2] that the DEDUCIBILITY problem (given a term t and a frame Φ, is t deducible in Φ?) reduces to STATEQ. As a consequence, the results of [5] imply that static equivalence is also undecidable for so-called *optimally-reducing* rewrite systems, a subclass of rewrite systems that have the finite-variant property [17].

Subterm convergent theories. Historically, complexity results for static equivalence only considered fixed theories [2,12], that is, the theory was not part of the input of the problem and its size was seen as a constant in the complexity analysis. This was consistent with most formalisms and verification tools at the time, which would not allow for user-defined theories and only consider a fixed set of cryptographic primitives, such as in the spi-calculus for example [4]. In particular fixed theories are considered in the following result:

Theorem 1 ([2]). *For all fixed subterm convergent theories* STATEQ *is* PTIME.

Generic PTIME-completeness results would make no sense when the theory is not part of the input, since the complexity may depend of it. Typically when using an empty theory the complexity changes:

Theorem 2 ([22]). *In the pure pi-calculus,* STATEQ *is* LOGSPACE.

However, in some sense, the PTIME bound is optimal since it is possible to provide a hardness result for a large class of fixed subterm convergent theories:

Theorem 3. *For all fixed theories containing symmetric encryption,* STATEQ *is* PTIME*-hard.*

Proof (Sketch). We proceed by reduction from HORNSAT. Let X the set of variables of a Horn formula $\varphi = C_1 \wedge \ldots \wedge C_n$, and k_x names for all $x \in X \cup \{\bot\}$. Then to each clause $C = x_1, \ldots, x_n \Rightarrow x$, $x \in X \cup \{\bot\}$ we associate the term

$$t_C = \mathsf{senc}(\ldots \mathsf{senc}(\mathsf{senc}(k_x, k_{x_1}), k_{x_2}), \ldots, k_{x_n})\,.$$

Putting k_x under several layers of encryption ensures that k_x is deducible if all the keys $k_{x_1}, \ldots, k_{x_n}$ are deducible as well. In particular $k_\bot$ is deducible from the terms $t_{C_1}, \ldots, t_{C_n}$ *iff* the formula φ is unsatisfiable. Hence given two constants 0,1, and $\Phi = \{\mathsf{ax}_1 \mapsto t_{C_1}, \ldots, \mathsf{ax}_n \mapsto t_{C_n}\}$, we have that the frames $\Phi \cup \{\mathsf{ax} \mapsto \mathsf{senc}(0, k_\bot)\}$ and $\Phi \cup \{\mathsf{ax} \mapsto \mathsf{senc}(1, k_\bot)\}$ are statically equivalent *iff* φ is satisfiable. □

However tools have improved since then and automated provers like KISS [28], YAPA [13] or FAST [31] are able to handle user-defined theories. It is therefore interesting today to refer to complexity analyses that account for the size of the rewrite system:

Theorem 4 ([22]). STATEQ *is coNP-complete for subterm convergent theories.*

Beyond subterm convergence. Although we are not aware of complexity results for the decision of static equivalence for classes larger than subterm theories, there exist decidability results. Some of the abovementioned tools, like KISS and YAPA, can actually handle most convergent rewriting system; but they naturally fail to terminate in general by undecidability of the problem. However it is proved in [28] that the termination of KISS is guaranteed for theories modelling blind signatures or trapdoor commitment (that are typically not subterm).

4 Complexity of dynamic equivalences (active attacker)

4.1 Equivalences

We expect security protocols to provide privacy against attackers that actively engage with the protocol. This can be modelled by behavioural equivalences, defining security as the indistinguishability of two instances of the protocol that differ on a privacy-sensitive attribute. There exist several candidate equivalences for modelling this notion of indistinguishability. We study two of them here and refer to [21] for details.

Trace equivalence. The first one is *trace equivalence*. Referring to the operational semantics described in Section 2, we call a *trace* t a sequence of transition steps

$$t = (A_0 \xrightarrow{\alpha_1} A_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} A_n)$$

If tr is the word obtained after removing unobservable actions (i.e. τ's) from the word $\alpha_1 \cdots \alpha_n$, the trace t may be written $A_0 \overset{\mathsf{tr}}{\Rightarrow} A_n$ instead. Processes P_0 and P_1 are then trace equivalent when for all traces $P_i \overset{\mathsf{tr}}{\Rightarrow} (P, \Phi)$, $i \in \{0,1\}$, there exists $P_{1-i} \overset{\mathsf{tr}}{\Rightarrow} (P', \Phi')$ such that the frames Φ and Φ' are statically equivalent. Automated verification of trace equivalence has been studied intensively for security protocols [7,20,21,22] and received strong tool support [17,18,23,32,33]. We refer to this problem as TRACEEQ:

INPUT: A theory, two processes.

QUESTION: Are the two processes trace equivalent?

Labelled bisimilarity. Some other tools prove stronger equivalences, like *observational equivalence* for PROVERIF [16,19] for example. There exist several flavours of more operational bisimulation-based properties but the one usually considered in security-protocol analysis is *labelled bisimilarity* because it coincides with observational equivalence in the applied pi-calculus [1]. Formally it is an early, weak bisimulation that additionally requires static equivalence at each step; that is, it is the largest symmetric binary relation $\approx$ on processes such that $A \approx B$ implies that (1) the frames of A and B are statically equivalent, and (2) for all actions α and all transitions $A \xrightarrow{\alpha} A'$, there exists a trace $B \overset{\alpha}{\Rightarrow} B'$ such that $A' \approx B'$. We refer to the following problem as BISIM:

INPUT: A theory, two processes.

QUESTION: Are the two processes labelled bisimilar?

4.2 Classical fragments of the calculus

In addition to the assumptions on the rewriting system (e.g. subterm convergence as in Section 3), there are several common restrictions made on the processes.

Conditionals and patterns. A typical restriction on conditionals is the class of *positive* processes that only contains trivial else branches [12,21,22]. In particular for succintness we often write $[u = v]P$ instead of $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ 0$.

When the rewrite system is constructor-destructor, some conditionals may also be encoded within inputs [26,27]. For that the syntax for inputs is generalised as $c(v).P$

where v is a term without destructors (but may contain variables), called a *pattern*. In terms of semantics, such inputs only accept terms that match the pattern, i.e. inputs t such that $t{\downarrow}= u\sigma{\downarrow}$ for some substitution σ, and then proceeds to execute $P\sigma$. In this paper, to ensure that protocols can be effectively implemented we require that it is possible to test with a sequence of positive conditionals that a term t matches the pattern u, and that all variables of u can be extracted by applying destructors to u. We thus define the *patterned* fragment to be the class of processes without conditionals but using pattern inputs, and where outputs do not contain destructor symbols; it is a subset of the positive fragment.

Ping pong protocols. These protocols [27,38,44] intuitively consist of an unbounded number of parallel processes receiving one message and sending a reply. Although the precise formalisms may differ from one work to another, the mechanisms at stake are essentially captured by processes $P = !P_1 \mid \cdots \mid !P_n$ where

$$P_i = c_i(x).\,[u_1^i = v_1^i] \cdots [u_{n_i}^i = v_{n_i}^i]\mathsf{new}\ k_1 \cdots \mathsf{new}\ k_{r_i}.\,\overline{c_i}\langle w_i\rangle$$

In particular ping-pong protocols are positive.

Simple processes. A common middleground in terms of expressivity and decidability is the class of simple processes, for example studied in [21,26]. Intuitively, they consist of a sequence of parallel processes that operate each on a distinct, public channel—including replicated processes that generate dynamically a fresh channel for each copy. Formally they are of the form

$$P_1 \mid \cdots \mid P_m \mid !^{ch}\, P_{m+1} \mid \cdots \mid !^{ch}\, P_n \qquad \text{where } !^{ch}\, P = !\ \mathsf{new}\ c_P.\,\overline{c'_P}\langle c_P\rangle.\,P$$

where each P_i does not contain any parallel operator ($|$) nor replication ($!$) and uses a unique, distinct communication channel c_{P_i}. Unlike ping pong protocols, each parallel process may input several messages and output messages that depend on several previous inputs. There exists a generalisation of simple processes called *determinate processes*, mentioned later in Section 5.

4.3 Complexity results: bounded fragment

The bounded fragment is a common restriction to study decidability, as removing replication bounds the length of traces. However, as the attacker still has an unbounded number of possibilities for generating inputs, the transition system still has infinite branching in general. Additional restrictions are also necessary on the cryptographic primitives (as static equivalence is undecidable in general). For subterm convergent, constructor-destructor theories for example:

Theorem 5 ([22]). TRACEEQ *and* BISIM *are decidable in co*NEXP *for subterm convergent constructor-destructor theories and bounded processes.*

We do not detail the decision procedures as they are quite involved. In a nutshell, they use a dedicated constraint-solving procedure to show that, whenever trace equivalence is violated, there exists an attack trace whose attacker-input terms are at most of

exponential size; in particular this shows non-equivalence to be decidable in NEXP. As before, we may also study the problem for fixed theories to investigate their influence on the complexity; typically with the empty theory:

Theorem 6 ([22]). *In the pure pi-calculus,* TRACEEQ *(resp.* BISIM*) is Π_2-complete (resp. PSPACE-complete) for bounded processes, and for bounded positive processes.*

However, unlike static equivalence, fixing the theory does not make it possible to obtain a bound that is better than the general one:

Theorem 7 ([22]). *There is a subterm-convergent constructor-destructor theory s.t.* TRACEEQ *and* BISIM *are coNEXP-hard for bounded positive processes.*

The theory in question [22] encodes binary trees and a couple of ad hoc functionalities. We show in the technical report [25] that, provided we discard the positivity requirement, the proof is possible with symmetric encryption and pairs only. This shows that the problem remains theoretically hard even with a minimal theory.

4.4 Complexity results: unbounded case

Equivalence is undecidable in general since the calculus is Turing-complete even for simple theories. For example, Hüttel [43] shows that Minsky's two counter machines can be simulated within the spi-calculus (and hence the applied pi-calculus with symmetric encryption only). It is not difficult to adapt the proof to a simulation using only a free symbol, i.e., a function symbol h of positive arity and an empty rewrite system. These two encodings can be performed within the *finite-control fragment*, typically not Turing-complete in the pure pi-calculus (i.e. without this free function symbol) [35].

Ping pong protocols. While equivalence is undecidable for ping-pong protocols [27,44] some decidability results exist under additional assumptions. For example [44] studies a problem that can be described in our model essentially as BISIM for ping-pong protocols with 2 participants or less (i.e. $n \leq 2$ in the definition). This is proved decidable under some model-specific assumptions which we do not detail here. We also mention a result for patterned ping-pong protocols (cf. Section 4.2) without a limit on the number of participants [27]. Given a constructor-destructor theory, a ping-pong protocol P is said to be *deterministic* when each P_i (using the same notations as in the definition) can be written under the form

$$P_i = c_i(u_i).\,\mathsf{new}\ k_1 \cdots \mathsf{new}\ k_{r_i}.\overline{c_i}\langle v_i \rangle$$

where $u_1, \ldots, u_n$ is a family of patterns verifying:
(1) *binding uniqueness*: for all i, u_i does not contain two different variables;
(2) *pattern determinism*: for all $i \neq j$, if u_i and u_j are unifiable then $c_i \neq c_j$.
There is an additional, minor restriction on the structure of u_i and v_i that is omitted here, and we refer to [27] for details.

Theorem 8 ([27]). *For a theory limited to randomised symmetric and asymmetric encryption and digital signature,* TRACEEQ *is decidable in primitive recursive time for deterministic ping-pong protocols.*

Decidability is obtained by a reduction of the problem to the language equivalence of deterministic pushdown automata, which is decidable in primitive recursive time. A complexity lower bound for this problem is open (beyond the PTIME-hardness inherited from static equivalence, recall Theorem 3).

Simple processes. We now study a decidability result for patterned simple processes [26]. In this work the theory is limited to symmetric encryption and pairs, and the processes must be *type compliant* and *acyclic*. Formalising the last two assumptions is quite technical and beyond the scope of this survey. We refer to the technical report [25] for more intuition and details. Compared to our other models, there is also a restriction to *atomic keys*, i.e. for all encryptions $\mathsf{senc}(u, v)$ in the process, v is either a constant, name or variable. This restriction is also enforced to attacker's recipes in the semantics by strenghtening the msg predicate.

Theorem 9 ([26]). *For a theory limited to pairs and symmetric encryption,* TRACEEQ *is coNEXP for patterned, simple, type-compliant, acyclic processes with atomic keys.*

The proof shows that equivalence of such (unbounded) processes is violated *iff* it is violated for a exponential number of sessions, and then uses a coNP decidability result in the bounded fragment [21]. However complexity was not the focus of [26] and the authors only claimed a triple exponential complexity for their procedure. Besides no lower bounds were investigated, but we proved that the problem was coNEXP-complete.

Theorem 10. *For a theory limited to pairs and symmetric encryption,* TRACEEQ *is coNEXP-hard for patterned, simple, type-compliant, acyclic processes with atomic keys.*

The reduction shares some similarities with the proof of coNEXP-hardness for trace equivalence of bounded processes (Theorem 7), compensating the more deterministic structure of simple processes by the use of replication.

Proof (Sketch). We proceed by reduction from SUCCINT 3SAT. This is a common NEXP-complete problem that, intuitively, is the equivalent of 3SAT for formulas of exponential size represented succinctly by boolean circuits. Formally a formula φ with 2^m clauses and 2^n variables $x_0, \ldots, x_{2^n-1}$ is encoded by a boolean circuit Γ : $\{0,1\}^{m+2} \to \{0,1\}^{n+1}$ in the following way. If $\varphi = \bigwedge_{i=0}^{2^m-1} \ell_i^1 \vee \ell_i^2 \vee \ell_i^3$ and $0 \leq i \leq 2^m - 1$ and $0 \leq j \leq 2$, we let x_k be the variable of the literal ℓ_i^{j+1} and b its negation bit; then $\Gamma(\bar{i}\,\bar{j}) = b\,\bar{k}$ where $\bar{i}, \bar{j}, \bar{k}$ are the respective binary representations of i, j, k. SUCCINT 3SAT is the problem of deciding, given a circuit Γ, whether the formula φ it encodes is satisfiable.

Let then φ be a formula with 2^m clauses and 2^n variables $x_0, \ldots, x_{2^n-1}$ and Γ a boolean circuit encoding this formula. We construct two simple, type-compliant, acyclic processes that are trace equivalent *iff* φ is unsatisfiable. Using pairs $\langle u, v\rangle$ we encode binary trees: a leaf is a non-pair value and, if u and v encode binary trees, $\langle u, v\rangle$ encodes the tree whose root has u and v as children. Given a term t, we build a process $P(t)$ behaving as follows:

(1) $P(t)$ first waits for an input x from the attacker. This term x is expected to be a binary tree of depth n with boolean leaves, modelling a valuation of φ (the i^{th} leaf of x being the valuation of x_i).

(2) The goal is to make $P(t)$ verify that this valuation satisfies φ; if the verification succeeds the process outputs t. Given two constants $\mathsf{0}$ and $\mathsf{1}$, $P(\mathsf{0})$ and $P(\mathsf{1})$ will thus be trace equivalent *iff* φ is unsatisfiable.
(3) However it is not possible to hardcode within a process of polynomial size the verification that the valuation encoded by x satisfies the 2^m clauses of φ. Hence we replicate a process that, given x, verifies one clause at a time. Intuitively, the attacker will guide the verification of the 2^m clauses of φ, and whenever the i^{th} clause has been successfully verified, the process reveals the binary representation of i (encrypted using a key unknown to the attacker).
(4) In particular, the attacker gets the encryption of all integers $0, \ldots, 2^m - 1$ only if she has successfully verified that the initial input x indeed encodes a valuation satisfying all clauses of φ. It then suffices to design a process that outputs t if the attacker is able to provide all such ciphertexts. This can be encoded by a replicated process that, upon receiving the encryption of two integers that differ only by their least significant bit, reveals the encryption of these integers with the least significant bit truncated. The verification ends when the encryption of the empty binary representation is revealed. □

5 Variations of the model

In this section we discuss a few variants of the model such as other notions of equivalence or different semantics of the process calculus.

Diff equivalence. The most well-known variant of equivalence properties in security protocols is *diff-equivalence*, different variants of which are proved by the state-of-the-art PROVERIF and TAMARIN. Intuitively, diff-equivalence can be seen as an analogue of trace equivalence where two equivalent traces are also required to follow the exact same execution flow. Consider for example the processes

$$P = \overline{c}\langle u\rangle \mid \overline{c}\langle v\rangle \qquad \text{and} \qquad Q = \overline{c}\langle u'\rangle \mid \overline{c}\langle v'\rangle\,.$$

Given the trace of P outputting u first and then v, a proof of trace equivalence could match it with either of the two traces of Q. However diff-equivalence only considers the trace of Q outputting u' first and then v'.

Theorem 11 ([16]). *If two processes are diff-equivalent then they are also labelled bisimilar (and therefore trace equivalent).*

The converse does not hold in general, which may lead to so-called *false attacks* (non-diff-equivalent processes that are, for example, trace equivalent). Regarding decidability and complexity, we call the following problem DIFFEQ:

INPUT: A theory, two processes.

QUESTION: Are the two processes diff equivalent?

Although undecidable in general, it is known to be decidable in the bounded, positive fragment [12]. More precisely it is shown that for all *fixed* subterm convergent theories, diff-equivalence is reducible to a coNP constraint-solving problem.

Theorem 12 ([12]). *For all fixed subterm convergent theories,* DIFFEQ *is coNP for positive bounded processes.*

It is also known that DIFFEQ is coNP-hard for a theory containing only a free binary symbol h [12]. A simple proof justifies that DIFFEQ is actually coNP-hard even for the empty theory and, hence, for any fixed theory:

Theorem 13. *In the pure pi-calculus,* DIFFEQ *is coNP-complete for positive bounded processes.*

Proof. By reduction from SAT we consider a formula $\varphi = \bigwedge_{i=1}^{m} C_i$ in CNF and $\boldsymbol{x} = x_1, \ldots, x_n$ its variables. For each clause C_i, let k_i be a name and define

$$\mathit{CheckSat}_i(\boldsymbol{x}) = [x_{i_1} = b_{i_1}]\overline{c}\langle k_i\rangle \mid \cdots \mid [x_{i_p} = b_{i_p}]\overline{c}\langle k_i\rangle$$

where $x_{i_1}, \ldots, x_{i_p}$ are the variables of C_i and $b_{i_1}, \ldots, b_{i_p}$ their respective negation bit. That is, at least one output of k_i is reachable in $\mathit{CheckSat}_i(\boldsymbol{x})$ if $\boldsymbol{x}$ is a valuation of φ that satisfies C_i. In particular if we define

$$\begin{aligned}\mathit{CheckSat} &= c(x_1).\ldots c(x_n).(\mathit{CheckSat}_1(\boldsymbol{x}) \mid \cdots \mid \mathit{CheckSat}_m(\boldsymbol{x}))\\ \mathit{Final}(t) &= c(y_1).[y_1 = k_1]\ldots c(y_m).[y_m = k_m]\ \overline{c}\langle t\rangle\end{aligned}$$

then for two distinct constants ok, ko, we have the processes $\mathit{CheckSat} \mid \mathit{Final}(\mathsf{ok})$ and $\mathit{CheckSat} \mid \mathit{Final}(\mathsf{ko})$ diff-equivalent *iff* φ is unsatisfiable.

As far as we know the complexity of diff-equivalence has only been studied for fixed theories. However the coNP-completeness [12] can be adapted to parametric theories; inspecting the proof we observe that (1) in the complexity bounds, the dependencies in the theory are polynomial and (2) the proof uses the fact that static equivalence is PTIME for fixed theories (Theorem 1) but the arguments still hold if we only assume static equivalence to be coNP.

Theorem 14. DIFFEQ *is coNP-complete for subterm convergent theories and positive bounded processes.*

Equivalence by session. We also briefly mention another equivalence, between diff-equivalence and trace equivalence (but incomparable with labelled bisimilarity) [24]. Known as *equivalence by session*, it was originally presented as a sound proof technique for trace equivalence in the bounded fragment, that was inducing less false attacks than diff-equivalence. We call this problem SESSEQ:

INPUT: A theory, two processes.

QUESTION: Are the two processes equivalent by session?

Surprisingly, despite practical improvements by order of magnitudes of the verification time compared to trace equivalence [24], this performance gap is not reflected in the theoretical, worst-case complexity. The same reduction as trace equivalence can indeed be used to prove equivalence by session coNEXP-hard. More details about the complexity of this problem can be found in the technical report [25].

Theorem 15. SESSEQ *is coNEXP-complete for constructor-destructor subterm convergent theories and positive bounded (resp. bounded) processes.*

The case of determinacy. We now mention the fragment of *determinate* processes, a generalisation of simple processes. In this fragment of the calculus, most of the studied equivalences coincide and their complexity also drops exponentially. This class has been investigated significantly [9,17,21,24] although several variants coexist in the literature, as discussed in [8]. For example the results of [9,24] hold for *action-determinate* processes, meaning that the processes never reach an intermediary state where two inputs (resp. outputs) on the same communication channel are executable in parallel; whereas a more permissive definition is used in [21]. There also exists a notion that is stricter than all of these, referred as *strong determinacy* [8]. A process is strongly determinate when (1) it does not contain private channels, (2) it is bounded, (3) all its syntactic subprocesses are strongly determinate, (4) in case the process is of the form $P \mid Q$ there exist no channels c such that both P and Q contain an input (resp. an output) on c. For example this process is action-determinate but not strongly-determinate:

$$\mathsf{if}\ a = b\ \mathsf{then}\ c(x)\ \mathsf{else}\ 0 \quad \mid \quad \mathsf{if}\ a = b\ \mathsf{then}\ 0\ \mathsf{else}\ c(x)\,.$$

Note in particular that bounded simple processes are strongly determinate.

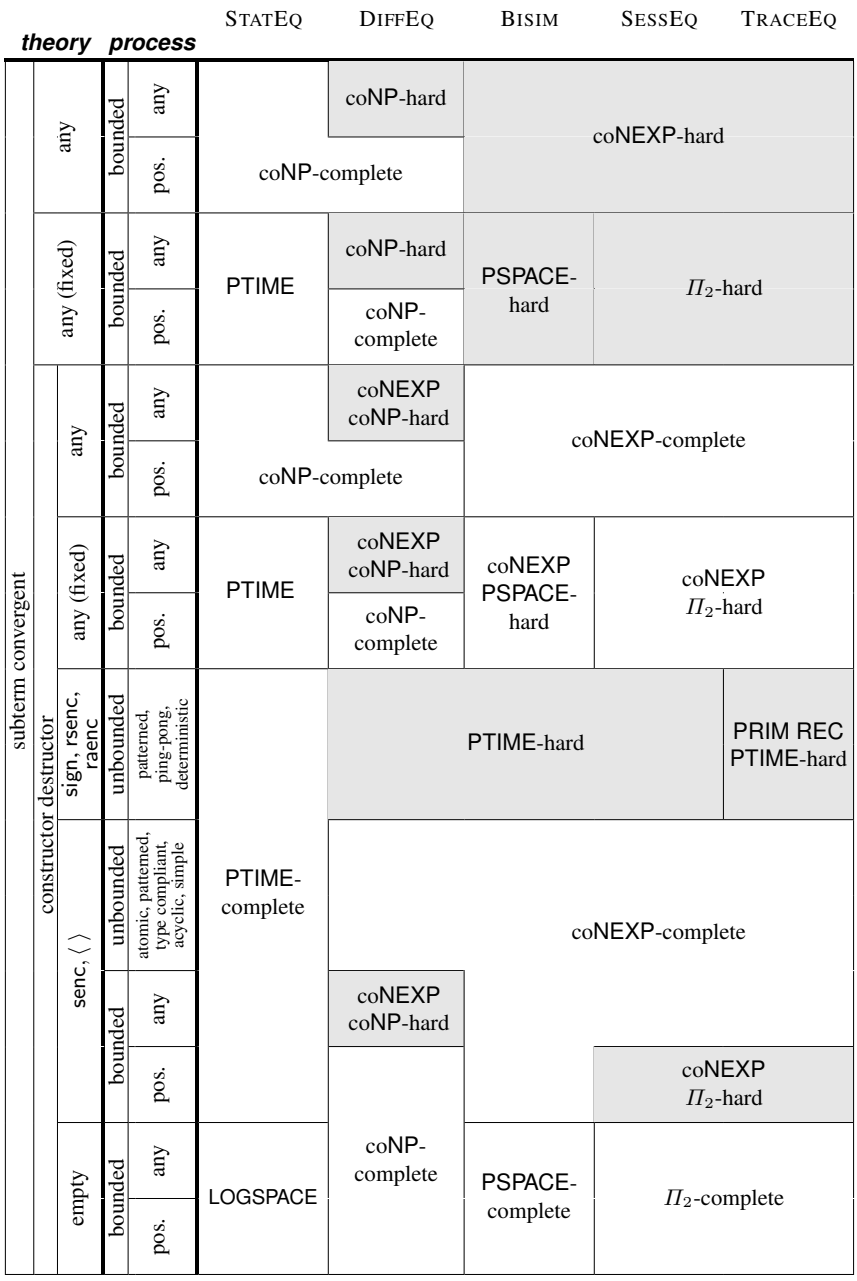
Theorem 16 ([21,24]). *Two labelled bisimilar (resp. equivalent by session) processes are trace equivalent. The converse is true for action-determinate processes.*

In [21] it is shown that, for bounded, simple, positive processes, the equivalence problem could be reduced to the same coNP constraint-solving problem mentioned in the paragraph on diff-equivalence. Their arguments can be generalised from simple to strongly-determinate processes in a straightforward manner; however it is not clear whether this would also be true for action-determinate processes or processes with else branches. In particular we obtain for this fragment:

Theorem 17 ([21]). TRACEEQ, BISIM *and* SESSEQ *are coNP-complete for subterm convergent theories and bounded, strongly determinate, positive processes. The coNP completeness also holds for all fixed subterm convergent theories.*

Variations of the communication model. Although all symbolic models rely on the same fundamental ideas, several variations exist in the semantics of communication, as noted in [8]. The differences lie in the modelling of silent communications between parallel processes. In the original semantics [1], called classical in [8], communications on a same public channel between parallel processes can either be an internal, synchronous, and silent action, or be intercepted by the attacker. This is also the semantics used in the popular PROVERIF tool [16]. On the contrary, the so-called *private* semantics only allows private, unintercepted communications on channels that are unknown to the attacker, modelling an attacker that continuously eavesdrops on the network (rather than an attacker that has the capability of eavesdropping any communication). The private semantics is actually used in tools such as TAMARIN [50] and AKISS [17] and also in the presentation of equivalence by session [24].

Table 1. Summary of the results. Colored cells indicate configurations with open problems. *All results for diff-equivalence also coincide with the results for trace equivalence, labelled bisimilarity, and equivalence by session for strongly-determinate processes.*



While both semantics are equivalent when it comes to reachability properties, they surprisingly happen to be incomparable for equivalence properties [8]. All the complexity results of this paper are with respect to the private semantics. Although we did not expand on studying all the variations of complexity induced by using different semantics, most of the analyses presented in this survey are robust to these changes. Indeed, all complexity results for the bounded fragment hold for both semantics. In the unbounded case, only the private semantics has been considered in the underlying models [26,27].

6 Summary of the results

Table 1 summarises the results of and highlights open questions (including a few results only detailed in the technical report [25] for space reasons). Cells for which the complexity results are not tight are colored in grey. For instance, for subterm-convergent constructor-destructor theories and bounded processes, DIFFEQ is known coNEXP and coNP-hard, but the precise complexity remains unknown. We also include in this table some complexity results with the theory seen as a constant of the problem (denoted as "fixed" in the *theory* columns). The corresponding cells contain bounds applying to *all* theories of the class; e.g. for BISIM of bounded processes, with fixed subterm-convergent constructor-destructor theories, the problem is decidable in coNEXP and PSPACE-hard; despite the gap between the two bounds, they are optimal since there exist theories for which the problem is either coNEXP-hard or PSPACE. Therefore this cell is not highlighted in grey. In our opinion the most interesting open questions are:

- Can upper bounds on constructor-destructor subterm convergent theories be lifted to more general subterm convergent theories?
- Without the positivity assumption, can we tighten the complexity for diff equivalence, and strongly determinate processes?

This last question might allow to better understand why strongly determinate processes benefit from optimisations that improve verification performance that much. Finally, as witnessed by the contrast between the high complexity of equivalence by session and its practical efficiency, worst-case complexity may not always be an adequate measure.

References

1. Abadi, M., Blanchet, B., Fournet, C.: The applied pi calculus: Mobile values, new names, and secure communication. Journal of the ACM (JACM) (2017)
2. Abadi, M., Cortier, V.: Deciding knowledge in security protocols under equational theories. Theoretical Computer Science (2006)
3. Abadi, M., Fournet, C.: Private authentication. Theoretcal Computer Science (2004)
4. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Information and computation (1999)
5. Anantharaman, S., Narendran, P., Rusinowitch, M.: Intruders with caps. In: International Conference on Rewriting Techniques and Applications (2007)
6. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Analysing unlinkability and anonymity using the applied pi calculus. In: IEEE Computer Security Foundations Symposium (CSF) (2010)
7. Arapinis, M., Cortier, V., Kremer, S.: When are three voters enough for privacy properties? In: European Symposium on Research in Computer Security (ESORICS) (2016)

8. Babel, K., Cheval, V., Kremer, S.: On the semantics of communications when verifying equivalence properties. Journal of Computer Security (2020)
9. Baelde, D., Delaune, S., Hirschi, L.: Partial order reduction for security protocols. In: International Conference on Concurrency Theory (CONCUR) (2015)
10. Basin, D.A., Cremers, C.: Know your enemy: Compromising adversaries in protocol analysis. ACM Transactions on Information and System Security (TISSEC) (2014)
11. Basin, D.A., Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R., Stettler, V.: A formal analysis of 5g authentication. In: ACM Conference on Computer and Communications Security (CCS) (2018)
12. Baudet, M.: Sécurité des protocoles cryptographiques: aspects logiques et calculatoires. Ph.D. thesis (2007)
13. Baudet, M., Cortier, V., Delaune, S.: YAPA: A generic tool for computing intruder knowledge. ACM Trans. Comput. Log. (2013)
14. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified models and reference implementations for the TLS 1.3 standard candidate. In: IEEE Symposium on Security and Privacy, (S&P) (2017)
15. Blanchet, B.: Modeling and verifying security protocols with the applied pi calculus and proverif. Foundations and Trends in Privacy and Security (2016)
16. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. The Journal of Logic and Algebraic Programming (2008)
17. Chadha, R., Cheval, V., Ciobâcă, Ş., Kremer, S.: Automated verification of equivalence properties of cryptographic protocols. ACM Transactions on Computational Logic (TOCL) (2016)
18. Cheval, V.: Apte: an algorithm for proving trace equivalence. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (2014)
19. Cheval, V., Blanchet, B.: Proving more observational equivalences with proverif. In: International Conference on Principles of Security and Trust (POST) (2013)
20. Cheval, V., Comon-Lundh, H., Delaune, S.: Trace equivalence decision: Negative tests and non-determinism. In: ACM conference on Computer and communications security (CCS) (2011)
21. Cheval, V., Cortier, V., Delaune, S.: Deciding equivalence-based properties using constraint solving. Theoretical Computer Science (2013)
22. Cheval, V., Kremer, S., Rakotonirina, I.: DEEPSEC: Deciding equivalence properties in security protocols theory and practice. In: IEEE Symposium on Security and Privacy (S&P) (2018)
23. Cheval, V., Kremer, S., Rakotonirina, I.: The deepsec prover. In: International Conference on Computer Aided Verification (CAV) (2018)
24. Cheval, V., Kremer, S., Rakotonirina, I.: Exploiting symmetries when proving equivalence properties for security protocols. In: ACM Conference on Computer and Communications Security (CCS) (2019)
25. Cheval, V., Kremer, S., Rakotonirina, I.: The hitchhiker's guide to decidability and complexity of equivalence properties in security protocols (technical report) (2020), available at https://hal.archives-ouvertes.fr/hal-02501577
26. Chrétien, R., Cortier, V., Delaune, S.: Decidability of trace equivalence for protocols with nonces. In: IEEE Computer Security Foundations Symposium (CSF) (2015)
27. Chrétien, R., Cortier, V., Delaune, S.: From security protocols to pushdown automata. ACM Transactions on Computational Logic (TOCL) (2015)
28. Ciobâcă, Ş., Delaune, S., Kremer, S.:
Computing knowledge in security protocols under convergent equational theories. In: International Conference on Automated Deduction (CADE) (2009)

29. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In: ACM Conference on Computer and Communications Security (CCS) (2018)
30. Comon, H., Cortier, V.: Tree automata with one memory set constraints and cryptographic protocols. Theoretical Computer Science (2005)
31. Conchinha, B., Basin, D.A., Caleiro, C.: Fast: an efficient decision procedure for deduction and static equivalence. In: International Conference on Rewriting Techniques and Applications (RTA) (2011)
32. Cortier, V., Dallon, A., Delaune, S.: Efficiently deciding equivalence for standard primitives and phases. In: European Symposium on Research in Computer Security (ESORICS) (2018)
33. Cortier, V., Grimm, N., Lallemand, J., Maffei, M.: A type system for privacy properties. In: ACM Conference on Computer and Communications Security (CCS) (2017)
34. Cremers, C., Horvat, M., Hoyland, J., Scott, S., van der Merwe, T.: A comprehensive symbolic analysis of TLS 1.3. In: ACMConference on Computer and Communications Security (CCS) (2017)
35. Dam, M.: On the decidability of process equivalences for the π-calculus. Theoretical Computer Science (1997)
36. Delaune, S., Hirschi, L.: A survey of symbolic methods for establishing equivalence-based properties in cryptographic protocols. Journal of Logical and Algebraic Methods in Programming (2017)
37. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security (2009)
38. Dolev, D., Yao, A.: On the security of public key protocols. In: Symposium on Foundations of Computer Science (FOCS) (1981)
39. Dolev, D., Even, S., Karp, R.M.: On the security of ping-pong protocols. Information and Control (1982)
40. Durgin, N.A., Lincoln, P., Mitchell, J.C.: Multiset rewriting and the complexity of bounded security protocols. Journal of Computer Security (2004)
41. Durgin, N.A., Lincoln, P., Mitchell, J.C., Scedrov, A.: Undecidability of bounded security protocols. In: Proc. Workshop on formal methods in security protocols (1999)
42. Filimonov, I., Horne, R., Mauw, S., Smith, Z.: Breaking unlinkability of the ICAO 9303 standard for e-passports using bisimilarity. In: European Symposium on Research in Computer Security (ESORICS) (2019)
43. Hüttel, H.: Deciding framed bisimilarity. Electronic Notes in Theoretical Computer Science (2003)
44. Hüttel, H., Srba, J.: Recursive ping-pong protocols. BRICS Report Series (2003)
45. Jacomme, C., Kremer, S.: An extensive formal analysis of multi-factor authentication protocols. In: IEEE Computer Security Foundations Symposium (CSF) (2018)
46. Kanovich, M.I., Kirigin, T.B., Nigam, V., Scedrov, A.: Bounded memory protocols. Computer Languages, Systems & Structures (2014)
47. Kobeissi, N., Bhargavan, K., Blanchet, B.: Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2017)
48. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I. Inf. Comput. (1992)
49. Rusinowitch, M., Turuani, M.: Protocol insecurity with a finite number of sessions, composed keys is NP-complete. Theoretical Computer Science (2003)
50. Schmidt, B., Meier, S., Cremers, C., Basin, D.: The TAMARIN prover for the symbolic analysis of security protocols. In: International Conference on Computer Aided Verification (CAV) (2013)